
9.2 Enabling and Disabling Interrupts

The processor services interrupts and exceptions only between the end of one instruction and the beginning of the next. When the repeat prefix is used to repeat a string instruction, interrupts and exceptions may occur between repetitions. Thus, operations on long strings do not delay interrupt response.

Certain conditions and flag settings cause the processor to inhibit certain interrupts and exceptions at instruction boundaries.

9.2.1 NMI Masks Further NMIs

While an NMI handler is executing, the processor ignores further interrupt signals at the NMI pin until the next IRET instruction is executed.

interrupt return

9.2.2 IF Masks INTR

The IF (interrupt-enable flag) controls the acceptance of external interrupts signalled via the INTR pin. When IF=0, INTR interrupts are inhibited; when IF=1, INTR interrupts are enabled. As with the other flag bits, the processor clears IF in response to a RESET signal. The instructions CLI and STI alter the setting of IF.

CLI (Clear Interrupt-Enable Flag) and STI (Set Interrupt-Enable Flag) explicitly alter IF (bit 9 in the flag register). These instructions may be executed only if CPL <= IOPL. A protection exception occurs if they are executed when

CPL > IOPL.

The IF is also affected implicitly by the following operations:

- The instruction PUSHF stores all flags, including IF, in the stack where they can be examined.
- Task switches and the instructions POPF and IRET load the flags register; therefore, they can be used to modify IF.
- Interrupts through interrupt gates automatically reset IF, disabling interrupts. (Interrupt gates are explained later in this chapter.)

9.2.3 RF Masks Debug Faults

The RF bit in EFLAGS controls the recognition of debug faults. This permits debug faults to be raised for a given instruction at most once, no matter how many times the instruction is restarted . (Refer to Chapter 12 for more information on debugging.)

9.2.4 MOV or POP to SS Masks Some Interrupts and Exceptions

Software that needs to change stack segments often uses a pair of instructions; for example:

```
MOV SS, AX
MOV ESP, StackTop
```

If an interrupt or exception is processed after SS has been changed but before ESP has received the corresponding change, the two parts of the stack pointer SS:ESP are inconsistent for the duration of the interrupt handler or exception handler.

To prevent this situation, the 80386, after both a MOV to SS and a POP to SS instruction, inhibits NMI, INTR, debug exceptions, and single-step traps at the instruction boundary following the instruction that changes SS. Some exceptions may still occur; namely, page fault and general protection fault. Always use the 80386 LSS instruction,

and the problem will not occur.

---

up: Chapter 9 -- Exceptions and Interrupts
prev: 9.1 Identifying Interrupts
next: 9.3 Priority Among Simultaneous Interrupts and Exceptions